Claim Amendments

Applicants have amended claim 1, 5, 57 and 59. Applicants set forth below a complete listing of the claims with the corresponding status indicated for each claim.

1. (Currently Amended) A method for providing a design test bench, the method comprising:

providing a single executable program adapted to create a primary thread and a secondary thread, the primary thread running VERILOG code on a ~~VERILOG~~ simulator that runs VERILOG code, the secondary thread running an interpreter that interprets a scripted routine comprising a user-defined call that is mapped to a VERILOG task;

providing a user-defined VERILOG function associated with the interpreter;

upon encountering the user-defined function, passing control from the ~~VERILOG~~ simulator to the interpreter to interpret the scripted routine; and

upon encountering the user-defined call, passing control from the interpreter to the ~~VERILOG~~ simulator.

2-4. (Cancelled).

5. (Currently Amended) The method of Claim 1, further comprising synchronizing the ~~VERILOG~~ simulator and the interpreter via semaphores.

6-56. (Cancelled).

57. (Currently Amended) The method of Claim 1, further comprising directly sharing variables between the ~~VERILOG~~ simulator and the scripted routines.

58. (Cancelled).

59. (Currently Amended) A method for providing a design test bench, the method comprising:

providing a single executable program adapted to create a primary thread and one or more secondary threads, the primary thread running VERILOG code on a ~~VERILOG~~ simulator that runs VERILOG code, each of the secondary threads running a corresponding interpreter that interprets an associated scripted routine, each scripted routine comprising an associated user-defined call that is mapped to a VERILOG task;

providing a plurality of user-defined VERILOG functions, each user-defined function associated with a corresponding interpreter;

upon encountering one of the user-defined functions, passing control from the ~~VERILOG~~ simulator to the corresponding interpreter to interpret the associated scripted routine; and

upon encountering one of the user-defined calls, passing control from the interpreter to the ~~VERILOG~~ simulator.